Software Engineering

System Models

Based on Software Engineering, 7th Edition by Ian Sommerville

Objectives

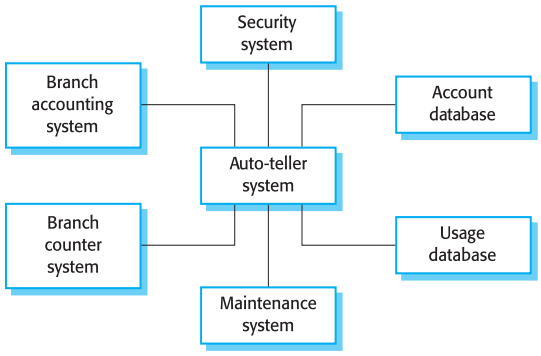
- To explain why the context of a system should be modeled as part of the RE process
- To describe behavioral modeling, data modeling and object modeling
- To introduce some of the notations used in the Unified Modeling Language (UML)
- To show how CASE workbenches support system modeling

System modeling

- System modeling helps the analyst to understand the functionality of the system and models are used to communicate with customers
- Different models present the system from different perspectives
  - External perspective showing the system's context or environment
  - Behavioral perspective showing the behavior of the system
  - Structural perspective showing the system or data architecture
- Model types
  - Data processing model showing how the data is processed at different stages
  - Composition model showing how entities are composed of other entities
  - Architectural model showing principal sub-systems
  - Classification model showing how entities have common characteristics
  - Stimulus/response model showing the system's reaction to events

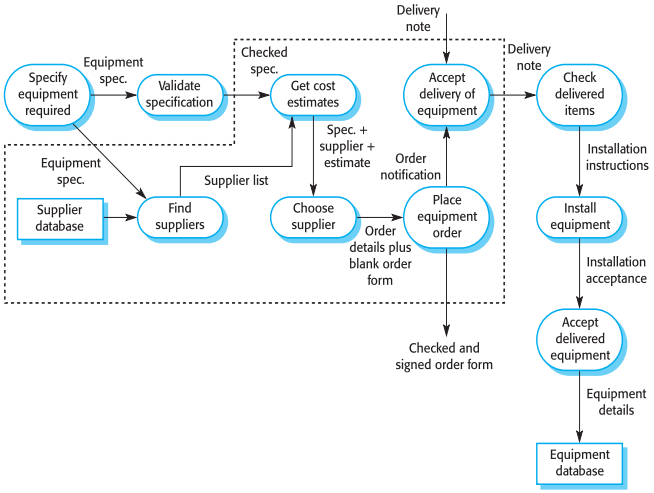
Context models

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organizational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.



Process models

- Process models show the overall process and the processes that are supported by the system.
- Data flow models may be used to show the processes and the flow of information from one process to another.



Stan Kurkovsky

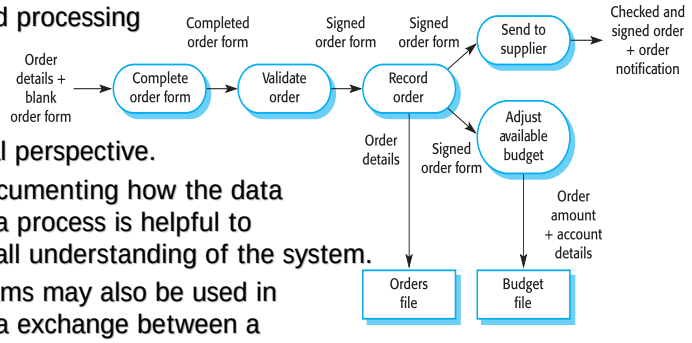
Behavioral models

- Behavioral models are used to describe the overall behavior of a system.
- Two types of behavioral model are:
  - Data processing models that show how data is processed as it moves through the system;
  - State machine models that show the systems response to events.
- These models show different perspectives so both of them are required to describe the system's behavior.

Stan Kurkovsky

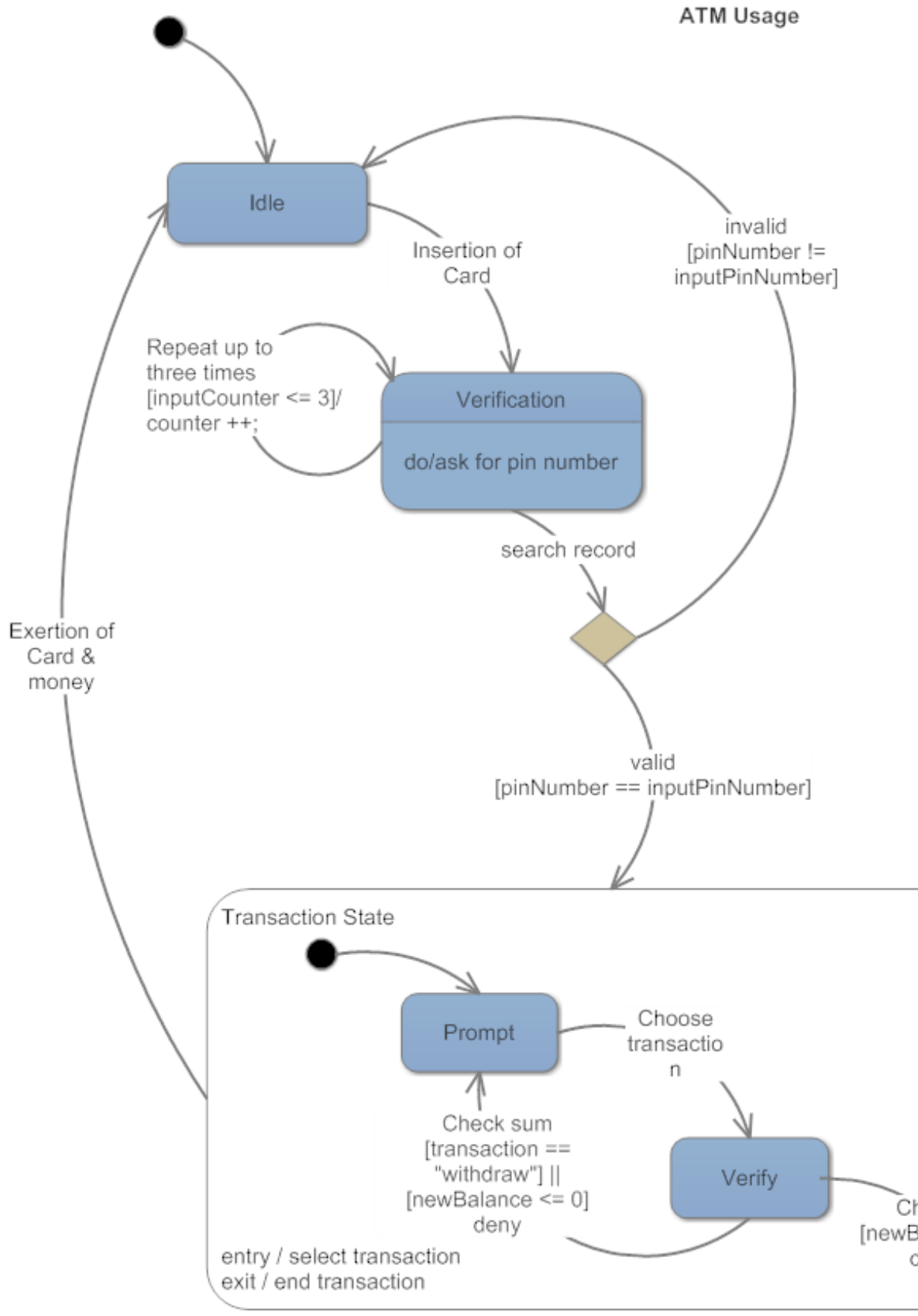
Data-processing models

- Data flow diagrams (DFDs) may be used to model the system's data processing.
- These show the processing steps as data flows through a system.
- DFDs are an intrinsic part of many analysis methods.
- Simple and intuitive notation that customers can understand.
- Show end-to-end processing of data.
- DFDs model the system from a functional perspective.
- Tracking and documenting how the data associated with a process is helpful to develop an overall understanding of the system.
- Data flow diagrams may also be used in showing the data exchange between a system and other systems in its environment.

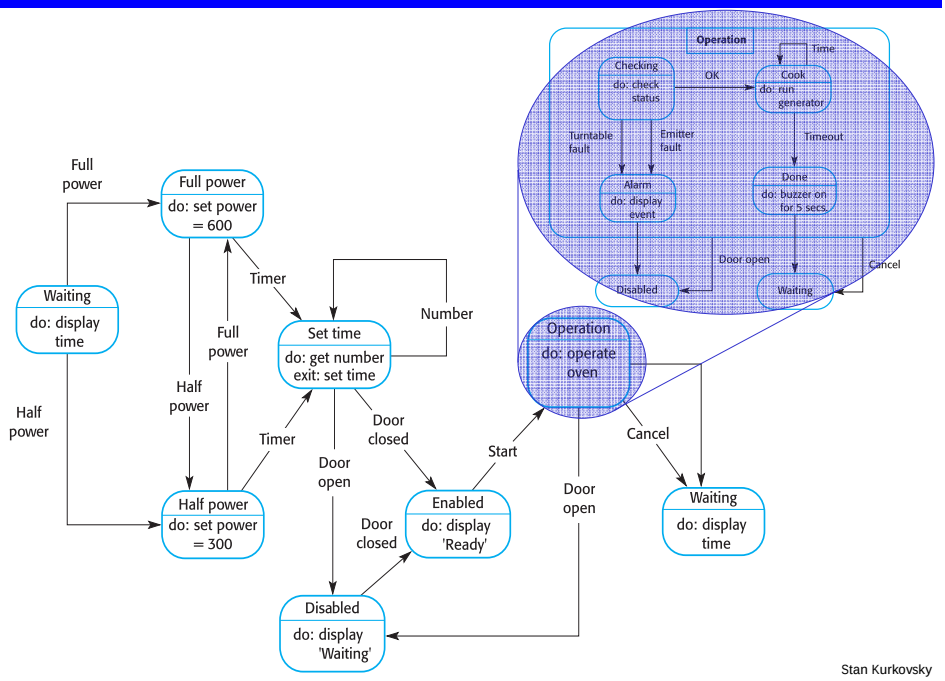


State machine models

- These model the behavior of the system in response to external and internal events.
- They show the system's responses to stimuli so are often used for modeling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another.
- Statecharts
  - An integral part of the UML and are used to represent state machine models.
  - Allow the decomposition of a model into sub-models (see following slide).
  - A brief description of the actions is included following the 'do' in each state.
  - Can be complemented by tables describing the states and the stimuli.


ATM Usage
Idle
Insertion of Card
invalid [pinNumber != inputPinNumber]
Repeat up to three times [inputCounter <= 3]/ counter ++;
Verification
do/ask for pin number
search record
Exertion of Card & money
valid [pinNumber == inputPinNumber]
Transaction State
Prompt
Choose transactio n
Check sum [transaction == "withdraw"] || [newBalance <= 0] deny
Verify
entry / select transaction
exit / end transaction

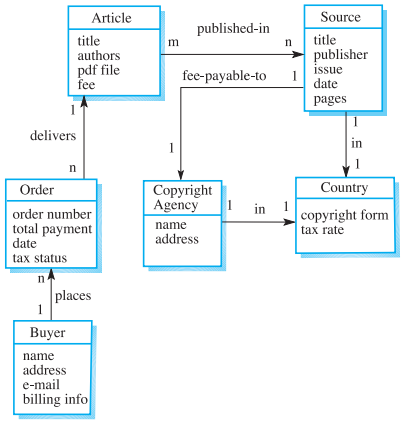
Microwave oven – state model



Stan Kurkovsky

Semantic data models

- Used to describe the logical structure of data processed by the system.
- An entity-relation-attribute model sets out the entities in the system, the relationships between these entities and the entity attributes
- Widely used in database design. Can readily be implemented using relational databases.
- No specific notation provided in the UML but objects and associations can be used.



Stan Kurkovsky

Data dictionaries

- Data dictionaries are lists of all of the names used in the system models. Descriptions of the entities, relationships and attributes are also included.
- Advantages
  - Support name management and avoid duplication;
  - Store of organizational knowledge linking analysis, design and implementation;
- Many CASE workbenches support data dictionaries.

Name	Description	Type	Date
Article	Details of the published article that may be ordered by people using LIBSYS.	Entity	30.12.2002
authors	The names of the authors of the article who may be due a share of the fee.	Attribute	30.12.2002
Buyer	The person or organisation that orders a copy of the article.	Entity	30.12.2002
fee-payable-to	A 1:1 relationship between Article and the Copyright Agency who should be paid the copyright fee.	Relation	29.12.2002
Address (Buyer)	The address of the buyer. This is used to any paper billing information that is required.	Attribute	31.12.2002

Stan Kurkovsky

Object models

- Object models describe the system in terms of object classes and their associations.
- An object class is an abstraction over a set of objects with common attributes and the services (operations) provided by each object.
- Various object models may be produced
  - Inheritance models;
  - Aggregation models;
  - Interaction models.
- Natural ways of reflecting the real-world entities manipulated by the system
- More abstract entities are more difficult to model using this approach
- Object class identification is recognized as a difficult process requiring a deep understanding of the application domain
- Object classes reflecting domain entities are reusable across systems

Stan Kurkovsky

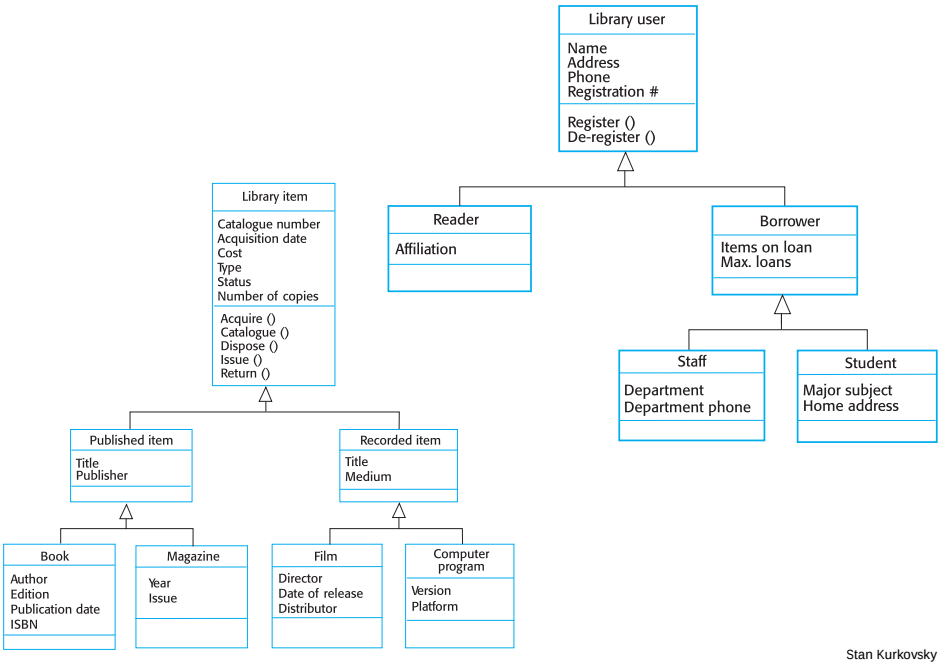
Inheritance models

- Organize the domain object classes into a hierarchy.
- Classes at the top of the hierarchy reflect the common features of all classes.
- Object classes inherit their attributes and services from one or more super-classes. these may then be specialized as necessary.
- Class hierarchy design can be a difficult process if duplication in different branches is to be avoided.

Object models and the UML

- The UML is a standard representation devised by the developers of widely used object-oriented analysis and design methods.
- It has become an effective standard for object-oriented modeling.
- Notation
  - Object classes are rectangles with the name at the top, attributes in the middle section and operations in the bottom section;
  - Relationships between object classes (known as associations) are shown as lines linking objects;
  - Inheritance is referred to as generalization and is shown ‘upwards’ rather than ‘downwards’ in a hierarchy.

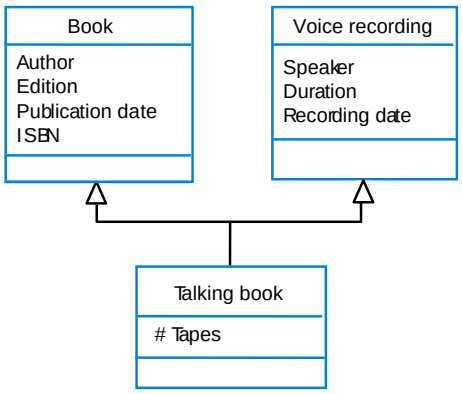
UML class diagrams



Stan Kurkovsky

Multiple inheritance

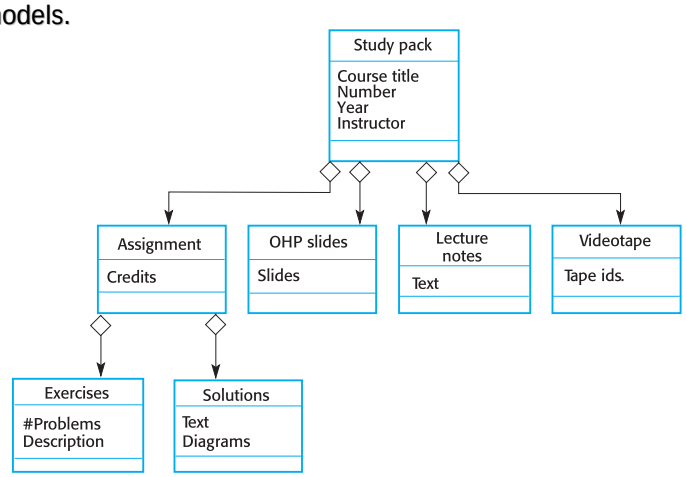
- Rather than inheriting the attributes and services from a single parent class, a system which supports multiple inheritance allows object classes to inherit from several super-classes.
- This can lead to semantic conflicts where attributes/services with the same name in different super-classes have different semantics.
- Multiple inheritance makes class hierarchy reorganization more complex.



Stan Kurkovsky

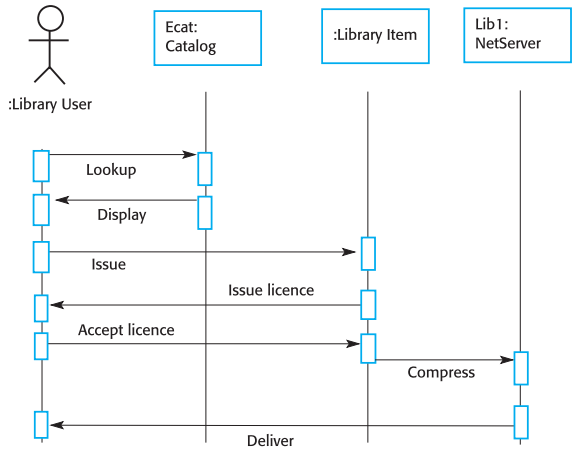
Object aggregation

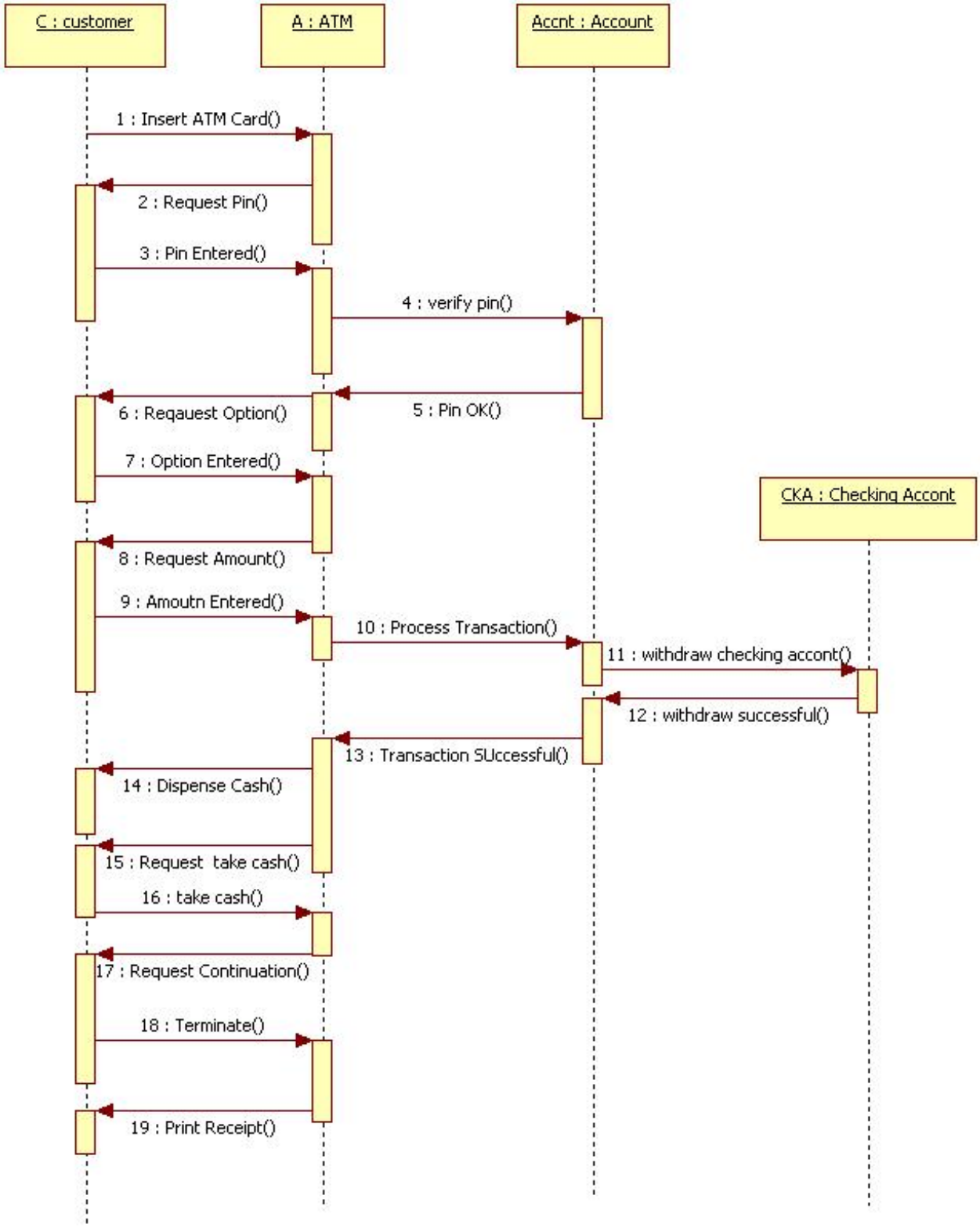
- An aggregation model shows how classes that are collections are composed of other classes.
- Aggregation models are similar to the part-of relationship in semantic data models.

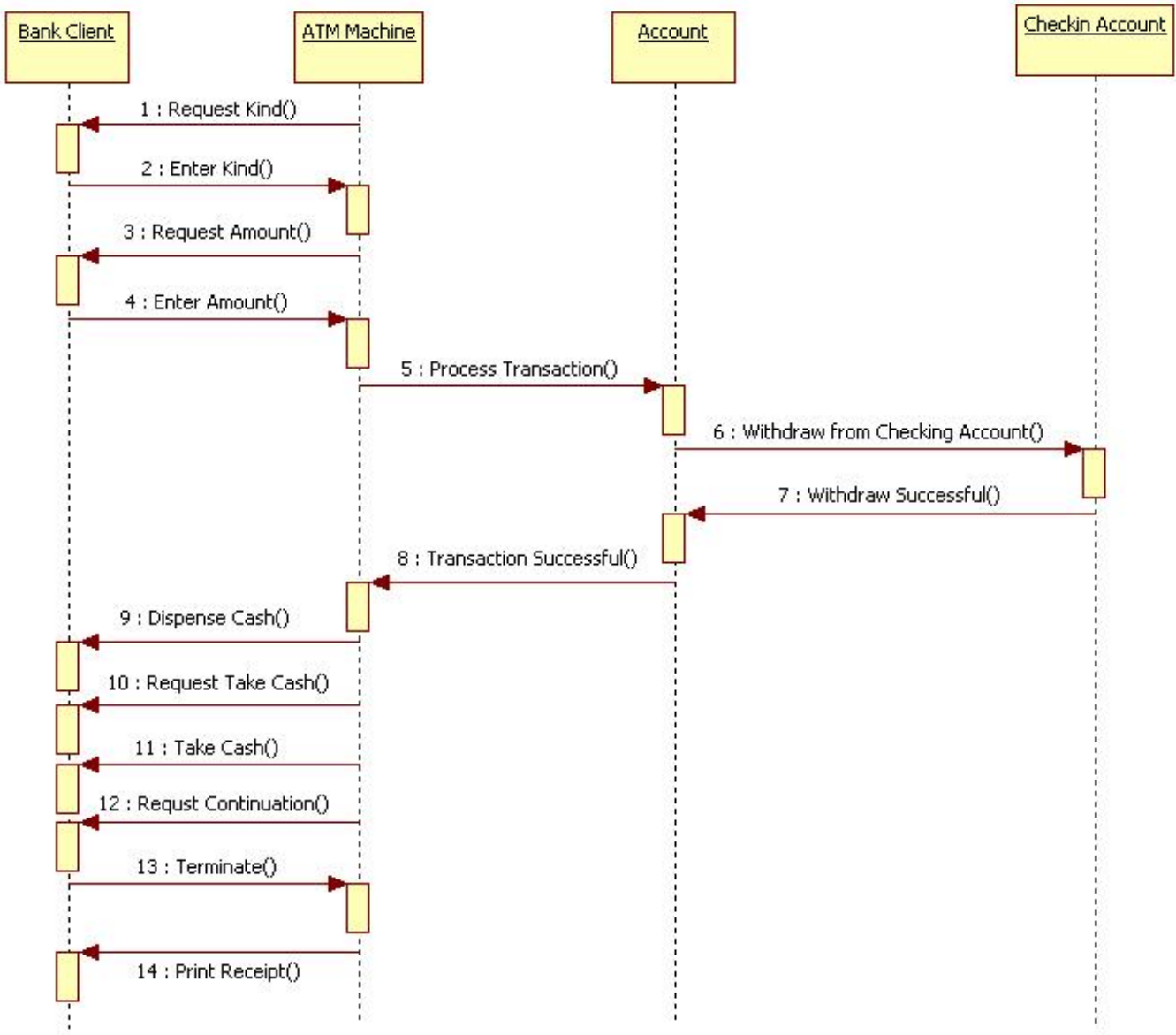


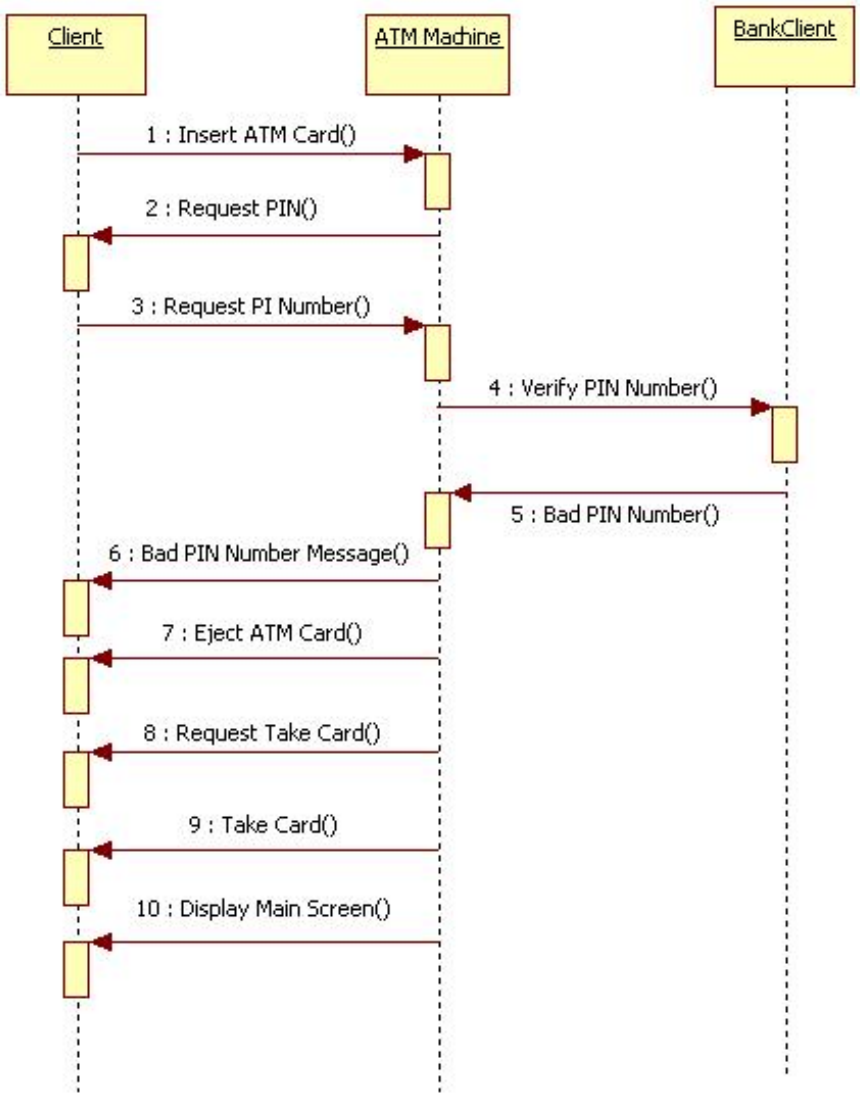
Object behavior modeling

- A behavioral model shows the interactions between objects to produce some particular system behavior that is specified as a use-case.
- Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects.




C : customer
A : ATM
Accnt : Account
CKA : Checking Accont
1 : Insert ATM Card()
2 : Request Pin()
3 : Pin Entered()
4 : verify pin()
5 : Pin OK()
6 : Reqauest Option()
7 : Option Entered()
8 : Request Amount()
9 : Amoutn Entered()
10 : Process Transaction()
11 : withdraw checking accont()
12 : withdraw successful()
13 : Transaction SUccessful()
14 : Dispense Cash()
15 : Request take cash()
16 : take cash()
17 : Request Continuation()
18 : Terminate()
19 : Print Receipt()


Bank Client
ATM Machine
Account
Checkin Account
1 : Request Kind()
2 : Enter Kind()
3 : Request Amount()
4 : Enter Amount()
5 : Process Transaction()
6 : Withdraw from Checking Account()
7 : Withdraw Successful()
8 : Transaction Successful()
9 : Dispense Cash()
10 : Request Take Cash()
11 : Take Cash()
12 : Requst Continuation()
13 : Terminate()
14 : Print Receipt()


Client
ATM Machine
BankClient
1 : Insert ATM Card()
2 : Request PIN()
3 : Request PI Number()
4 : Verify PIN Number()
5 : Bad PIN Number()
6 : Bad PIN Number Message()
7 : Eject ATM Card()
8 : Request Take Card()
9 : Take Card()
10 : Display Main Screen()

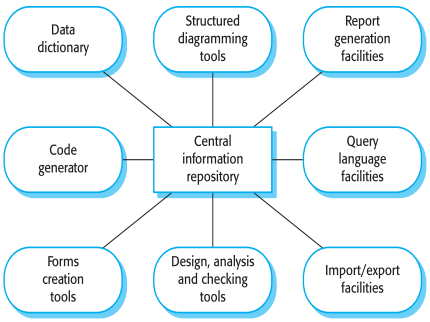
Structured methods

- Structured methods incorporate system modeling as an inherent part of the method.
- Methods define a set of models, a process for deriving these models and rules and guidelines that should apply to the models.
- CASE tools support system modeling as part of a structured method.

- Weaknesses
  - They do not model non-functional system requirements.
  - They do not usually include information about whether a method is appropriate for a given problem.
  - The may produce too much documentation.
  - The system models are sometimes too detailed and difficult for users to understand.

CASE workbenches

- A coherent set of tools that is designed to support related software process activities such as analysis, design or testing.
- Analysis and design workbenches support system modeling during both requirements engineering and system design.
- These workbenches may support a specific design method or may provide support for a creating several different types of system model.



- Workbench components
  - Diagram editors
  - Model analysis and checking tools
  - Repository and associated query language
  - Data dictionary
  - Report definition and generation tools
  - Forms definition tools
  - Import/export translators
  - Code generation tools

Key points

- A model is an abstract system view. Complementary types of model provide different system information.
- Context models show the position of a system in its environment with other systems and processes.
- Data flow models may be used to model the data processing in a system.
- State machine models model the system's behavior in response to internal or external events
- Semantic data models describe the logical structure of data which is imported to or exported by the systems.
- Object models describe logical system entities, their classification and aggregation.
- Sequence models show the interactions between actors and the system objects that they use.
- Structured methods provide a framework for developing system models.